Welcome

Bill Bartlett
billb.503@gmail.com
503.504.5644
2FA - Why?

- **Avoid Moral Hazard**
  - “Moral hazard is a situation in which one party gets involved in a risky event knowing that it is protected against the risk and the other party will incur the cost.”
  - Provide Opt-In Choices

- **Avoid Legal Hazard**
  - Lawsuits from parties affected by data breach

- **Cyber Insurance**
  - Lower premiums
  - Does nothing for those who suffer Identity Theft

- **Current Technology**
  - OSS / easy
  - BYOD / inexpensive
2FA – How?

- 2FA: Time-Based OTP
  - Google Authenticator App

- 2FA: One-Time Password (OTP)
  - Yubico

- 2FA: FIDO Universal 2nd Factor (U2F)
  - PKI Digital Signature

- 2FA: CLEF
  - PKI Digital Signature
## 2FA - Comparison

<table>
<thead>
<tr>
<th>2FA Possession</th>
<th>2FA Biometric</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Experience UX</td>
<td>Not ready for prime time</td>
</tr>
<tr>
<td>Security</td>
<td>Analog to Digital conversion</td>
</tr>
<tr>
<td>Implementation</td>
<td>Slow algorithms</td>
</tr>
<tr>
<td>Cost</td>
<td>Accuracy</td>
</tr>
<tr>
<td></td>
<td>- False negatives</td>
</tr>
<tr>
<td></td>
<td>- False positives</td>
</tr>
<tr>
<td></td>
<td>- Fuzzy logic</td>
</tr>
<tr>
<td></td>
<td>Sensors not universally ready</td>
</tr>
<tr>
<td></td>
<td>Costly</td>
</tr>
<tr>
<td></td>
<td>User resistance to biometric databases</td>
</tr>
</tbody>
</table>
Where we have been
Passwords

- Security by Knowledge
  - Complexity
    - 1 number
    - 1 non-alpha
    - Mixed case
    - Minimum length
  - Aging
    - Renew every x days
  - No reuse

- Convenience
  - Memory
  - Password Reset via eMail
Password Insecurity

- Static values
- Autofill
  - Password managers
  - Browser cache
- User habits
  - Reuse
  - Written lists
  - Patterns
  - Password Fatigue
  - Shoulder surfing
- Attacks
  - Hashing rainbow tables
  - Cross site hacks
  - Social engineering
Password Implementation

- Implementation
  - Clear text
    - Bad dog!
  - Encrypt/Decrypt
    - Secure Key management
  - Hash
  - Hash with Salt

- User Provision
  - PinHash VARCHAR(64)
  - UUID VARCHAR(36)

- Best practice
  1. Verify new password (2x)
  2. Hash(Password + UUID)
  3. Throttling retries after 3x
Password Implementation

Register

```csharp
// Save the PINhash
    Hash.HashType.SHA256);
dbUser.UpdatePassword();
```

Verify

```csharp
// Validate the Pin
if (!Hash.CheckHash(model.Password + Hash.GetSalt(dbUser.UUID),
    dbUser.PINhash,
    Hash.HashType.SHA256))
    {...handle fail...}
```
Where we are going
Google TOTP

- Security by Possession
- BYOD
- Authenticator Mobile App
  - iOS, Android
  - Offline
  - $0.00
- Simple registration
  - Scan QR code
- Awkward verification
  - Read 6 digit code on mobile device
  - Enter 6 digit code on login page
Google TOTP Security

- 80 bit symmetrical user key
  - Mobile Device
  - Web server database
    - Encryption needed

- Challenge/Response Algorithm
  - # Time segments (30 second increments)
  - Hashed, Truncated, Digitized

- Risks
  - Predictable Challenge sequence
  - Clock drift
    - Verify 3 time offset values (-1, 0, +1)
  - The keys must be protected
  - If the master key is compromised, the whole database is compromised
Google TOTP Implementation

**Register**
- Generate random key (80 bits)
- Provision for user
- Display QR bar code
- App scans the QR code

**Verify**
- Verify TOTP Enabled for user
- Collect TOTP1 from user via keyboard
- Retrieve key from Provision
- Generate TOTP2 on web server
- Verify TOTP1 = TOTP2

**User Provision**
- IdUser INT (fkey)
- Key VARCHAR(32)
- Enabled BIT
Google TOTP Implementation

Register

```c
// Get base32 value of 80 bits
datamodel.TOTP = totp.GenerateGoogleKey();
```

Verify

```c
// verify
if (totp.VerifyGoogleOTP(model.TOTP, model.OTP))
    model.Result = "Success!";
else
    model.Result = "Fail!";
```
Yubikey OTP

- Security by Possession
- Touch interface
- Keyboard emulation
- USB
- NFC
- $30 - $80
Yubikey OTP Security

- 12 digit serial # + 32 digit OTP
- Dynamic values
- One time usage
- Cloud Verify
- No drivers
- No batteries
- No moving parts
- No codes to remember
- Impenetrable to malware of any kind
- 128 bit encryption
- Endorsed by Google
Yubikey OTP Implementation

- Register
  - Collect OTP
  - Verify OTP with Web API
  - Provision in database

- Verify
  - Collect OTP
  - Retrieve provision
  - Verify User
  - Verify Enabled
  - Verify OTP with Web API

- Device Provision
  - IDbUser INT (fkey)
  - SerialNumber VARCHAR(12)
  - Enabled BIT
Yubikey OTP Implementation

// Verify OTP
if (!ycloud.IsVerify(model.YubicoSerialNumber)) {...handle fail...}

// Decode the serial number
string ysn = model.YubicoSerialNumber.ToUpper().Substring(0, 12);
ysn = ycloud.ToDecimal(ycloud.ToHex(ysn)).ToString();

// Provision
...
dbUserToken.Insert();

// Verify OTP
if (!ycloud.IsVerify(model.YubicoSerialNumber)) {...handle fail...}

// Decode the serial#
YubicoCloud ycloud = new YubicoCloud();
string ysn = model.YubicoSerialNumber.ToUpper().Substring(0, 12);
ysn = ycloud.ToDecimal(ycloud.ToHex(ysn)).ToString();

// Retrieve the User/Token
dbUserToken.SerialNumber = ysn;
dbUserToken.Find();
FIDO U2F

- Security by Possession
- Any U2F compliant device
- USB
- Touch interface
- Challenge/Response
- PKI Private/Public key pairs
- $10 - $20

Register
- User provides a tag/name for the device
- Touch the device to mint a PKI key pair

Verify
- User selects the device from a list
- Touch the device to verify a digital signature
FIDO U2F Security

- Public Key Infrastructure
  - Private/Public key pairs
  - Elliptical Curve Digital Signature Algorithm (ECDSA)
  - Private key derived in the device from a key handle
  - Public key stored in the web server database

- Challenge/Response Algorithm
  - Random challenge value is generated on the web server
  - Challenge value is signed by the private key on the device
  - The private key never leaves the device
  - The signature (~70 bytes) is returned to the web server
  - The public key is retrieved to verify the signature
  - Keys are tied to a domain

- Universal
  - Users can have multiple devices
  - Same device can be used on multiple domains

- Limitation
  - Google Chrome version 38+ with U2F extension
FIDO U2F Implementation

Register
- Assemble the Registration data
  - Random challenge
  - Domain ID
  - Version
- JS: window.u2f.register( registration )
- Retrieve Client Data
  - Challenge
  - Origin
  - Type
- Retrieve Registration Data
  - Public Key
  - Key Handle
  - X509 Certificate
  - Digital Signature
- Verify Digital Signature
  - X509 Public Key
- Provision device

Device Provision
- IdUser INT (fkey)
- Device name VARCHAR(32)
- Key handle VARBINARY(128)
- Public key VARBINARY(128)
- Enabled BIT
FIDO U2F Implementation

- Verify Digital Signature
  - Assemble the Sign data
    - Random challenge
    - Domain ID
    - Version
    - Key handle
  - JS: window.u2f.sign( signdata )
- Retrieve Client Data
  - Challenge
  - Origin
  - Type
- Retrieve Signature Data
  - User Present
  - Counter
  - Digital Signature
- Verify Digital Signature
  - With provisioned Public Key
- Verify Counter
FIDO U2F Implementation

```javascript
window.u2f.register([[
    challenge: "@Model.Challenge",
    version: "@Model.Version",
    appid: "@Model.AppId"
]], [], function (data) {...}
registerU2F.ParseClientData();
registerU2F.ParseRegistrationData();
registerU2F.CheckSignature();
if (registerU2F.ErrorFree) {...handle success...} else {...handle failure...}

window.u2f.sign([[
    challenge: "@Model.Challenge",
    version: "@Model.Version",
    appid: "@Model.AppId",
    keyHandle: "@Model.KeyHandle"
]], function (data) {...}
signU2F.ParseClientData();
signU2F.ParseSignatureData();
signU2F.CheckSignature();
if (signU2F.ErrorFree) {...handle success...} else {...handle failure...}
```
CLEF PKI

- Security by Possession
- Free SmartPhone App
- PKI Private/Public key pairs
  - Private key in your phone
  - Public key at Clef.com
- Register
  - Install SmartPhone App
  - Register with GetClef.com
- Verify
  - Friction-less 2FA from the future
CLEF PKI Security

- **Public Key Infrastructure**
  - Private/Public key pairs
  - 2048 bit PKI
  - Private key stored on the user SmartPhone
  - Public key stored at Clef
  - Out-of-band authentication is extremely secure
  - No Security credentials are stored on your website

- **Challenge/Response Digital Signature**
  - Receive barcode challenge from Clef via SmartPhone camera
  - Clef app signs the challenge with Private key
  - SmartPhone delivers digital signature out of band to Clef
  - Clef verifies the signature with Public key
  - Clef posts a token to your website
  - You post the token back to Clef and receive a user ID#
  - You verify the user ID# is registered to the user logging in
CLEF PKI Implementation

- **Register**
  - SmartPhone is registered with GetClef.com
  - User registers with your website
  - Digital Signature authentication occurs with the Clef wave barcode
  - A token is received from Clef
  - The token is posted to Clef
  - A User ID# is received from Clef

- **Device Provision**
  - IdUser INT (fkey)
CLEF PKI Implementation

```html
<div class="form-group form-horizontal">
  @Html.Label("Clef Authentication", htmlAttributes: new { @class = "control-label col-md-2" })
  <div class="col-md-10">
    <script type="text/javascript"
      class="clef-button"
      src="@Model.ClefURLJS"
      data-state="@Model.ClefDataState"
      data-app-id="@Model.ClefAppId"
      data-redirect-url="@Model.ClefURLCallback"
      data-color="blue"
      data-style="button"
      data-type="register">
    </script>
  </div>
</div>
```
CLEF PKI Implementation

```html
<div class="form-group form-horizontal">
    @Html.Label("Clef Authentication", htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-9">
        @if (Model.success)
        {
            <script type="text/javascript"
                class="clef-button"
                src="@Model.ClefURLJS"
                data-state="@Model.ClefDataState"
                data-app-id="@Model.ClefAppId"
                data-redirect-url="@Model.ClefURLCallback"
                data-color="blue"
                data-style="button"
                data-type="login">
                </script>
        }
        else
        {
            @TempData["Message"]
        }
    </div>
</div>
```
# 2FA Summary

<table>
<thead>
<tr>
<th>User Experience</th>
<th>Google TOTP</th>
<th>Yubico OTP</th>
<th>FIDO U2F</th>
<th>Clef.com</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form Factor</td>
<td>Mobile device</td>
<td>Hardware token</td>
<td>Hardware token</td>
<td>Mobile device</td>
</tr>
<tr>
<td>User Interface</td>
<td>App</td>
<td>USB / NFC</td>
<td>USB</td>
<td>App</td>
</tr>
<tr>
<td>User Action</td>
<td>6 digit transcribe</td>
<td>Touch activated</td>
<td>Touch activated</td>
<td>Point camera at the computer</td>
</tr>
<tr>
<td>Algorithm</td>
<td>Symmetric Encryption</td>
<td>Symmetric Encryption</td>
<td>PKI Digital Signature</td>
<td>PKI Digital Signature</td>
</tr>
</tbody>
</table>
## 2FA Summary

<table>
<thead>
<tr>
<th>Security</th>
<th>Google TOTP</th>
<th>Yubico OTP</th>
<th>FIDO U2F</th>
<th>CLef</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Algorithm</strong></td>
<td>Public algorithm</td>
<td>Vendor specific algorithm</td>
<td>ECDSA</td>
<td>DSA</td>
</tr>
<tr>
<td><strong>Challenge Value</strong></td>
<td>Time period (predictable)</td>
<td>Random Counters</td>
<td>Random Challenge</td>
<td>Vendor</td>
</tr>
<tr>
<td><strong>Method</strong></td>
<td>Calculate TOTP on web server and compare to user value</td>
<td>Cloud Verification Web API</td>
<td>(a) Digital Signature verify on web server (b) Verify counters are incrementing</td>
<td>(a) Out of band challenge and response (b) Async result notification</td>
</tr>
<tr>
<td><strong>Weakness</strong></td>
<td>90 second window for replay, Clock drift</td>
<td>No replay</td>
<td>Requires Chrome browser</td>
<td></td>
</tr>
</tbody>
</table>

- **2FA Summary**
  - **Security**: Methods used for two-factor authentication
  - **Algorithm**: Public vs. Vendor specific algorithms
  - **Challenge Value**: Time period or random counters
  - **Method**: Calculation and verification methods
  - **Weakness**: Security vulnerabilities
## 2FA Summary

<table>
<thead>
<tr>
<th>Integration</th>
<th>Google TOTP</th>
<th>Yubico OTP</th>
<th>FIDO U2F</th>
<th>Clef</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key Management</td>
<td>Keys are vulnerable in web database unless encrypted</td>
<td>Keys are stored in Yubico cloud</td>
<td>Public keys stored on web database, Private keys never leave device</td>
<td>Public key stored at Clef Private stored in Smartphone</td>
</tr>
<tr>
<td>Complexity</td>
<td>Easy</td>
<td>Moderate</td>
<td>Difficult</td>
<td>Difficult</td>
</tr>
<tr>
<td>Browser</td>
<td>Any browser</td>
<td>Any browser</td>
<td>FIDO extension required (Chrome 38)</td>
<td>Any browser</td>
</tr>
<tr>
<td>BYOD</td>
<td>$0 (SmartPhone)</td>
<td>$30 - $80</td>
<td>$10 - $20</td>
<td>$0 (SmartPhone)</td>
</tr>
<tr>
<td>Server Cost</td>
<td>$0</td>
<td>$0</td>
<td>$0</td>
<td>Volume pricing</td>
</tr>
</tbody>
</table>
2FA Summary

- Best Practices
  - Self Service
    - Opt-In
    - Enroll
    - Repudiate
  - User Notification
    - Email
    - SMS
DEMO

Give Identity Theft the Finger